НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

.

Дипломна робота

на тему: 3D-моделювання будівель

Студент групи ТР-62 Фахріян Денис Фаріборзович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник роботи доцент, к.т.н. Шаповалова С.І. _____
(вчені ступінь та звання, прізвище, ініціали) (підпис)

Кількість сторінок _____

Кількість ілюстрацій _____

Київ – 2020

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

зі спеціальності 122 Комп’ютерні науки та інформаційні технології
за спеціалізацією Геометричне моделювання в інформаційних системах
на тему **3Д-моделювання будівель**

Виконав: студент **4** курсу, групи **ТР-62**
Фахріян Денис Фаріборзович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник **доцент, к.т.н. Шаповалова С.І.** _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент **ст. викладач, к.т.н. Рачинський А.Ю.** _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

зі спеціальності 122 Комп’ютерні науки та інформаційні технології
за спеціалізацією Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ

на дипломну роботу студенту

Фахріяну Денису Фаріборзовичу
(прізвище, ім’я, по батькові)

1. Тема роботи **3Д-моделювання будівель**

керівник роботи **Шаповалова Світлана Ігорівна, к.т.н. , доцент**
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 201__р. № __

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи __мова програмування Blueprint, двигун Unreal Engine 4, 3D-редактор Blender, графічні редактори Substance Painter та Adobe Photoshop, системи Chaos Destruction System та Niagara

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) __проаналізувати специфіку розробки програмного коду на візуальній мові програмування Blueprint в двигуні Unreal Engine 4, розробити 3D-макети будівель для випробувань, запрограмувати логіку детонації модульного об’єкту, провести ряд експериментів, оптимізувати програму для використання на різних

пристроях.

5. Перелік ілюстративного матеріалу

6. Дата видачі завдання "_11_" _жовтня_____2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	01.10.2019 р.	
2.	Вивчення та аналіз задачі	15.11.2019 р.	
3.	Розробка архітектури та загальної структури системи	10.01.2020 р.	
4.	Розробка структур окремих підсистем	13.02.2020 р.	
5.	Програмна реалізація системи	15.03.2020 р.	
6.	Оформлення пояснювальної записки	25.05.2020 р.	
7.	Захист програмного продукту	11.05.2020 р.	
8.	Передзахист	08.06.2020 р.	
9.	Захист	15.06.2020 р.	

Студент _____ Фахріян Д.Ф.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шаповалова С.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Записка містить 52 сторінку, 40 рисунків, 3 додатки та 2 посилання.

Мета роботи – створити систему детонації модульної будівлі, яка може бути інтегрована та використовуватись в будь-якому ігровому програмному забезпеченні з наданням користувачу системи можливості заміни об’єктів сцени та параметрів вибуху.

Вибір двигуна Unreal Engine 4 для роботи, було зроблено, оскільки даний двигун дозволяє розробляти динамічні кросплатформні системи для кінофільмів, ігор та додатків, використовуючи вбудовані в двигун модулі та можливість програмувати на візуальній мові Blueprint. А також є популярним інструментом серед розробників та має власний маркетплейс з іншими об’єктами та системами, що можна використовувати з даним програмним забезпеченням.

У результаті було розроблено систему детонації модульного об’єкту з гнучким набором параметрів, які можна персоналізувати під особисті потреби користувачів системи.

Ключові слова: двигун UE4, кросплатформний програмний застосунок, система детонації, система модульної збірки об’єкту, редагування системи.

ABSTRACT

The note contains 52 pages, 40 figures, 3 appendices and 2 links.

The main purpose of the work is to create a modular building detonation system that can be integrated and used in any gaming software, giving the system user the ability to replace scene objects and explosion parameters.

The choice of the Unreal Engine 4 engine to work with was made because this engine allows to develop dynamic cross-platform systems for movies, games and applications, using built-in modules and the ability to program in visual language

Blueprint. It is also a popular tool among developers and has its own marketplace with other objects and systems that can be used with this software.

As a result, a modular object detonation system was developed with a flexible set of parameters that can be customized to the personal needs of system users.

Keywords: UE4 engine, cross-platform software application, detonation system modular object assembly system, system editing.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

VFX (visual effects) – візуальна система частинок, що використовується у комп'ютерній графіці для створення спецефектів.

LOD (level of details) – рівень деталізації моделі та/або матеріалів, що накладені на модель, в залежності від відстані від камери до об’єкту.

UE4 (Unreal Engine 4) – ігровий двигун компанії Epic Games.

BP (blueprint) – візуальна мова програмування. Також може бути визначений, як один із елементів системи.

FPS (frames per second) – одиниця виміру кількості кадрів за секунду в комп’ютерній графіці.

ВСТУП

В сучасному світі перед розробниками ігор все частіше постає питання оптимізації графіки та передання реалістичності картинки. Зокрема, створення таких моделей та спеціальних ефектів, які б могли безперешкодно використовуватись на мобільних пристроях. Сьогодні саме ринок мобільних додатків має найбільший потенціал. Наразі майже кожна друга людина світу має смартфон з доступом до Apple Store (під операційну систему iOS) та Google Play Market (під ОС Android). Це означає, що цей ринок можна потенційно збільшити ще вдвічі. Аналізуючи дані за період з 2012 по 2020рр., представлені аудиторською компанією Newzoo, що займається дослідженнями ринків ігор, можна зробити припущення, що у цьому, 2020-му році, прибутки компаній-розробників та незалежних інді-розробників ігор та додатків зростуть до відмітки в 189 мільярдів доларів США. Зі зростом попиту почався і зріст кількості розробників, що призвело до жорсткої конкуренції в цій ніші. Саме тому, кожному з них потрібно знайти свою конкурентну перевагу та робити максимальний акцент на ній. В даній дипломній роботі представлена одна з ряду можливих конкурентних переваг – графіка та візуалізація спецефектів руйнування.

З появою жорсткіших вимог до якості графіки мобільних ігор, великою кількістю об’єктів та режиму мультиплеєр (онлайн режим з можливістю підключення до серверу нових гравців), додатках із вбудованою 3D-візуалізацією та мобільних кінофільмів, сценарії яких побудовані за принципом вільного вибору сюжетних поворотів зі сторони геймерів. Виникло питання ефективного використання ресурсів мобільного пристрою: операційну пам’ять, кеш та характеристики телефона. Це дозволить максимально реалістичного передавати ефект, що приблизить графіку мобільних ігор до графіки десктопних версій, а в

майбутньому, можливо, й замінить їх. Саме тому створення прикладного ПЗ для подальшого використання в розробках ігор або візуалізації імітаційного моделювання **є актуальним та має практичну значущість**.

Другим **актуальним** питанням серед розробників ігор постав пошук готових систем, які можна кастомізувати під свої потреби. Що значно економить час та зазвичай не потребує дуже глибоких знань від розробників ПЗ. Це дозволяє ефективно розподілити навантаження на спеціалістів різних рівнів підготовки.

Саме до такого програмного забезпечення відноситься плагін симуляції руйнування будівель, який може використовуватись на персональних комп’ютерах, а в перспективі має бути пристосованим і під інші пристрої: планшети, смартфони, VR-окуляри, тобто стати кроссплатформенную. Завдяки цьому він може з легкістю використовуватися в комп’ютерних іграх, де більш важлива швидкість обробки фізичних об’єктів, а ніж реалізм.

1 ПОСТАНОВКА ЗАДАЧІ З ДЕТОНАЦІЇ БУДІВЛІ

Використовуючи відомості та теоретичні матеріали, знайдені в мережі Інтернет та в документаціях до використаного програмного забезпечення, розробити систему симуляції детонації модульних об’єктів за допомогою точок вибуху.

В програмне забезпечення має бути закладено готові моделі конструкцій будівель та логіку вибуху. В програмній системі можуть використовуватися як базові конструкції будівель, що представлені в даній дипломній роботі, так і інші об’єкти, які можуть бути розроблені та імпортовані в двигун користувачем, при відповідних налаштуваннях параметрів.

Метою роботи є створення програмного забезпечення руйнування будівель (Destructible Buildings System) для застосування на різних пристроях та з можливістю використання моделей користувача.

Для досягнення цієї мети необхідно виконати такі **завдання**:

1. Виокремити параметри моделювання ситуації та поставити задачі моделювання 3D-об’єктів для сцени. Визначити засоби їх реалізації;
2. Визначити формули та алгоритм розв’язання задачі фізичної симуляції детонації;
3. Розробити програмне забезпечення з можливістю заміни базових моделей на користувацькі;
4. Провести ряд обчислювальних експериментів по виявленню помилок та оптимізації навантаження програмного продукту на відеокарту;
5. Створити інструкцію по роботі з системою та інтеграцією 3D-моделей користувача в неї.

За постановкою задачі на сцені, яка містить будинок та навколишню територію, має бути розташовані одна чи кілька точок детонації для організації вибуху. В системі має бути передбачено використання епіцентрів вибуху з однаковими або відмінними характеристиками та наступними параметрами детонуючого пристрою:

- радіус дії;
- місцезнаходження відносно будівлі;
- напрямок вибухової хвилі;
- таймер до вибуху;
- сила.

Програмний продукт має бути інтуїтивно зрозумілим для його використання з іншими системними модулями користувача.

2 ЗАСОБИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В даному розділі представлені програмні засоби, на основі яких було розроблено плагін:

- Blender 3D – для моделювання та нанесення текстур;
- Substance Painter та Adobe Photoshop – для створення текстур та масок для 3D-об’єктів;
- Unreal Engine – використана як основна платформа для відтворення симуляції;
- C++ / Blueprint – для написання логіки роботи системи;
- VFX – для створення спецефектів вибуху.

2.1 3D-редактор Blender 3D

Blender 3D - це вільне, безкоштовне, open-source, GPL програмне забезпечення для створення 3D-графіки, анімації, відео, ігор та іншого. У блендері знаходиться величезна кількість інструментів для моделювання та анімації, потужний відео редактор, а також він може використовуватись як незалежний ігровий двигун.

2.2 Графічні редактори Substance Painter та Adobe Photoshop

Adobe Photoshop - програмне забезпечення, яке широко використовується для редагування растрових зображень, графічного дизайну та цифрового мистецтва. Він використовує шари, щоб забезпечити глибину та гнучкість у процесі проектування та редагування.

Substance Painter - 3D інструмент малювання. Його можна порівняти з 3D-версією Photoshop Adobe для роботи з цифровим живописом.

2.3 Ігровий двигун Unreal Engine 4

Unreal Engine 4 - ігровий двигун, створений компанією Epic Games. Unreal Engine - це повний набір інструментів розробки для тих, хто працює з технологіями в реальному часі. Від візуалізації дизайну та кінематографічного досвіду до високоякісних ігор на ПК, консолі, мобільних пристроях, VR та AR.

2.4 Мова програмування C++

C++ - високорівнева мова програмування, яка використовується на обчислювальних машинах та має широкий функціонал. Також на мобільних пристроях може бути використана як низькорівнева мова. Якщо порівнювати C++ з блюпринтами, вона є більш гнучкою і дозволяє маніпулювати більшою кількістю властивостей та параметрів об’єкту.

2.5 Візуальна псевдомова Blueprint

Blueprint - це швидкий і гнучкий спосіб збирання базової інфраструктури проекту. Це дозволяє в мінімальні терміни запустити робочий процес і швидко приступити до розробки програмного коду. Блюпринти мають вигляд візуального програмування, та містять ноди.

Нода - це події, виклики функцій, операції управління потоком, змінні, та інші об’єкти, які використовуються у графі візуалізації логіки для визначення функціональності конкретного графа та блюпринта, що його містить.

2.6 Спецефекти VFX

VFX (англ. Visual Effects) - система візуалізації спецефектів, що використовуєть у кінематографі та розробці ігор. Основним елементом системи є частинки, що дозволяють художникам створювати візуальні ефекти, починаючи від диму, іскорок та вогню до масштабних вибухів, цунамі та інших.

2.7 VFX-система Niagara

Система Niagara VFX - це один із двох інструментів, за допомогою яких можна створювати та коригувати візуальні ефекти (VFX) всередині Unreal Engine 4 (UE4). До Niagara основним способом створення та редагування візуальних ефектів в UE4 було використання Cascade. Хоча в Niagara є багато тих самих методів маніпулювання частинками, які пропонує Cascade, спосіб взаємодії та створення візуальних ефектів з Niagara відрізняється.

2.8 Внутрішня система фізики та руйнування Chaos Destruction System

Chaos Destruction System - нова високоефективна система фізики та руйнування Unreal Engine. Завдяки даній системі користувачі можуть в режимі реального часу досягати кінематографічної якості в сценах з масштабними руйнування.

2.9 Фізичні моделі

Фізична модель – це система, що дозволяє здійснювати фізичне моделювання, заміщаючи реальний фізичний процес, подібним до нього, але з частковим або повним симулюванням.

На рисунку 2.1 продемонстровано взаємозв’язок між компонентами, що використовуються у системі.

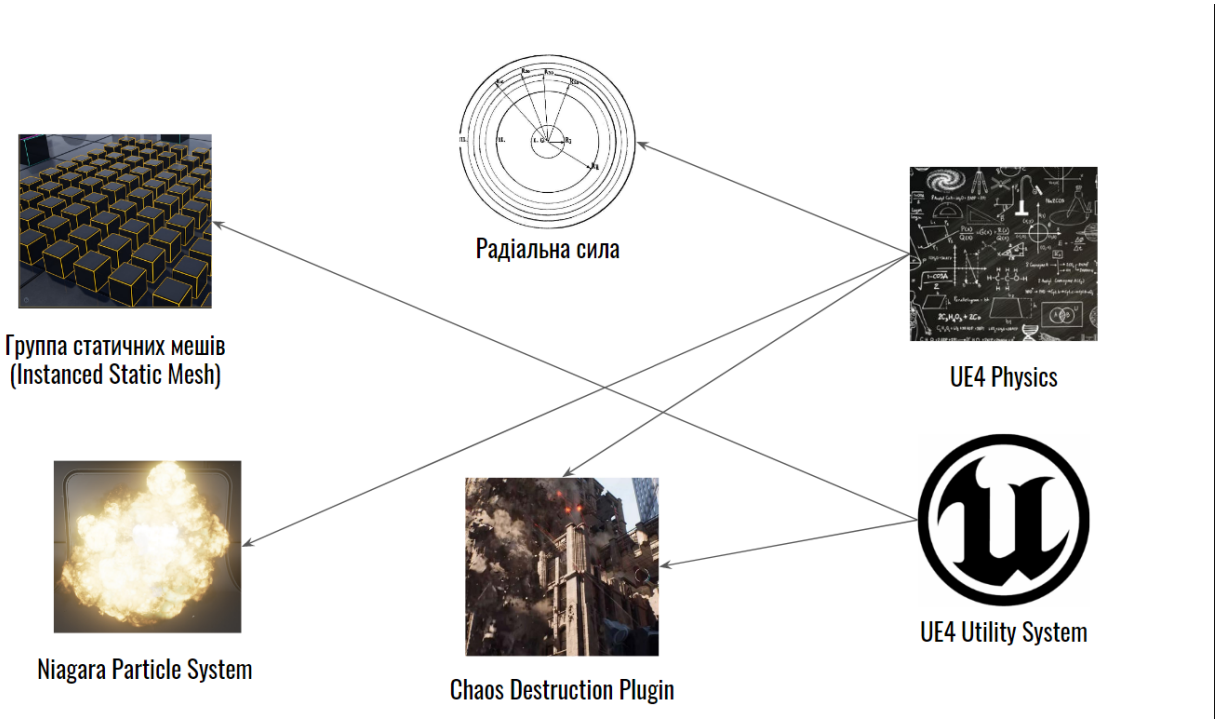


Рисунок 2.1 - Зв'язок компонентів системи

В системі використано ряд програмних засобів та зображено зв’язок між компонентами плагіну для виконання даної роботи.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Визначення моделей фізичного світу та їх оптимізація

Задля економії ресурсів пристрою, на якому буде використовуватися система руйнування будівель, в її програмне забезпечення закладено спрощену фізичну модель, а також не відбуваються попередні розрахунки параметрів за спрощеними фізичним формулам, та без попередніх підрахунків параметрів руйнування. Підсумкова калькуляція проводиться без урахування впливу зовнішніх сил, щільності матеріалу конструкції моделі будівлі, а також без урахування зміщеного центра мас кожного об’єкту. Вважається, що кожен об'єкт має центр мас в його геометричному центрі. Згасання швидкості частини конструкції після вибуху не розраховується, а задається сталим для кожного об’єкту окремо. Кількість уламків, їх розміри та розташування не вираховуються або створюються, а задаються константою для кожного об’єкту до початку вибуху. Тощо. Лінійне та кутове демпфірування також задається стало для кожного об’єкту.

Таблиця 3.1 відображає існуючу фізичну модель, та її оптимальна версія, що використовується у даній системі в ігровому двигуні Unreal Engine 4.

Таблиця 3.1 – Фізична модель руйнування будинку

Формула	Призначення	Вигляд	Спрощення	Посилання
Надлишковий тиск вибуху	Основною вражаючою дією вибухових речовин є ударна хвиля. Тому для визначення вражаючої дії вибухової речовини необхідно розрахувати надлишковий тиск вибуху	$D_p = P - P_0$ де P - тиск на фронті ударної хвилі; P_0 - тиск незбуреного повітря - атмосферний тиск (101кПа)	Вираховується в двигуні, як точка радіальної сили	(1.1)
Приведений радіус зони вибуху	Розрахунок величини надлишкового тиску р проводиться в два етапи. На першому етапі знаходиться приведений радіус зони вибуху за формулою	$\bar{R} = \frac{R}{\sqrt[3]{2KMT_\epsilon}}$ де R - відстань від центру вибуху, м; M - маса заряду, кг;	Радіус встановлюється вручну користувачем системи	(1.2)

Продовження таблиці 3.1 - Фізична модель руйнування будинку

Формула	Призначення	Вигляд	Спрощення	Посилання
		К - коефіцієнт, що враховує характер підстильної поверхні; Т - тротиловий еквівалент вибухової речовини.		

Також система використовує систему часточок, так звані Particles або VFX. Це дозволяє доповнити фізичний вибух, візуальними елементами, для придання вибуху реалістичності та кінематографічності, не навантажуючи при цьому обчислювальні машини великими обсягами зайвої інформації для проведення такого роду процесів. Більшість обчислень VFX виконуються на відеокарті, що сприяє прискоренню процесу відображення та зменшує навантаження на процесор, задля проведення підрахунків поведінки фізичних об’єктів на сцені.

Вибух здійснюється з огляду на фізичні закони природи та відповідні математичні моделі, задля здобуття максимально реалістичного ефекту.

Враховуються такі характеристики уламків:

- вага уламку;
- місцезнаходження уламку відповідно інших об’єктів вибуху;
- область колізії;
- згасання швидкості розльоту після вибуху;
- лінійне демпфірування.

Колізія - є основою для того, як Unreal Engine 4 обробляє зіткнення та викид променів під час роботи. Кожен об'єкт, який може зіткнутися, отримує Тип об'єкта та ряд відповідей, які визначають, як він взаємодіє, будуть усі інші типи об’єктів. Коли відбувається подія зіткнення або перекриття, обидва (або всі) об'єкти, що беруть участь, можуть бути встановлені таким чином, щоб впливати або впливати на блокування, накладення чи ігнорування один одного.

Лінійне демпфірування - використовують щоб уповільнити фізичні тіла, імітувати атмосферне перетягування або додати опір шарніру, для фізичних тіл та обмежень фізики доступні дві властивості: лінійне демпфування та кутове затухання. Лінійне затухання контролює, наскільки фізичне тіло чи обмеження чинить опір перекладу, тоді як кутове затухання керує тим, наскільки вони чинять опір обертанню.

З урахуванням цих характеристик кожна окрема частина при детонації будівлі на кожному певному проміжку часу обчислюється за такими характеристиками:

- швидкість польоту;
- траєкторія польоту;

3.2 Визначення елементів конструкції будівлі. Моделювання 3D-об’єктів.

Вхідною інформацією для моделювання була вибрана будівля та встановлені в ній точки вибуху. Для реалізації цієї задачі, в першу чергу, був обраний 3D-редактор Blender 3D.

Першим кроком потрібно було змоделювати будівлю, поділити її на частини (модульне моделювання об’єктів) та зібрати всі ці модулі в єдиний об’єкт:

- фундамент;
- опорна балка;
- дерев’яна підлога;

- стіна фундаменту;
- стіна з віконною рамою;
- кутова стіна з віконною рамою;
- суцільна стіна;
- суцільна кутова стіна;
- стіна з дверним отвором;
- дерев’яна перегородка;
- сходи;
- двері;
- віконниця;
- віконне скло;
- декоративна верхівка вікна;
- каркас для даху;
- черепиця;
- верхівка черепиці;
- димохід.

На рисунку 3.1 зображено частини елементів конструкції однієї із будівель, які представлені в виді статичних мешів. При необхідності ці меші можна корегувати, а саме, змінювати їх:

- розташування;
- обертання;
- розмір;
- нанесений матеріал.

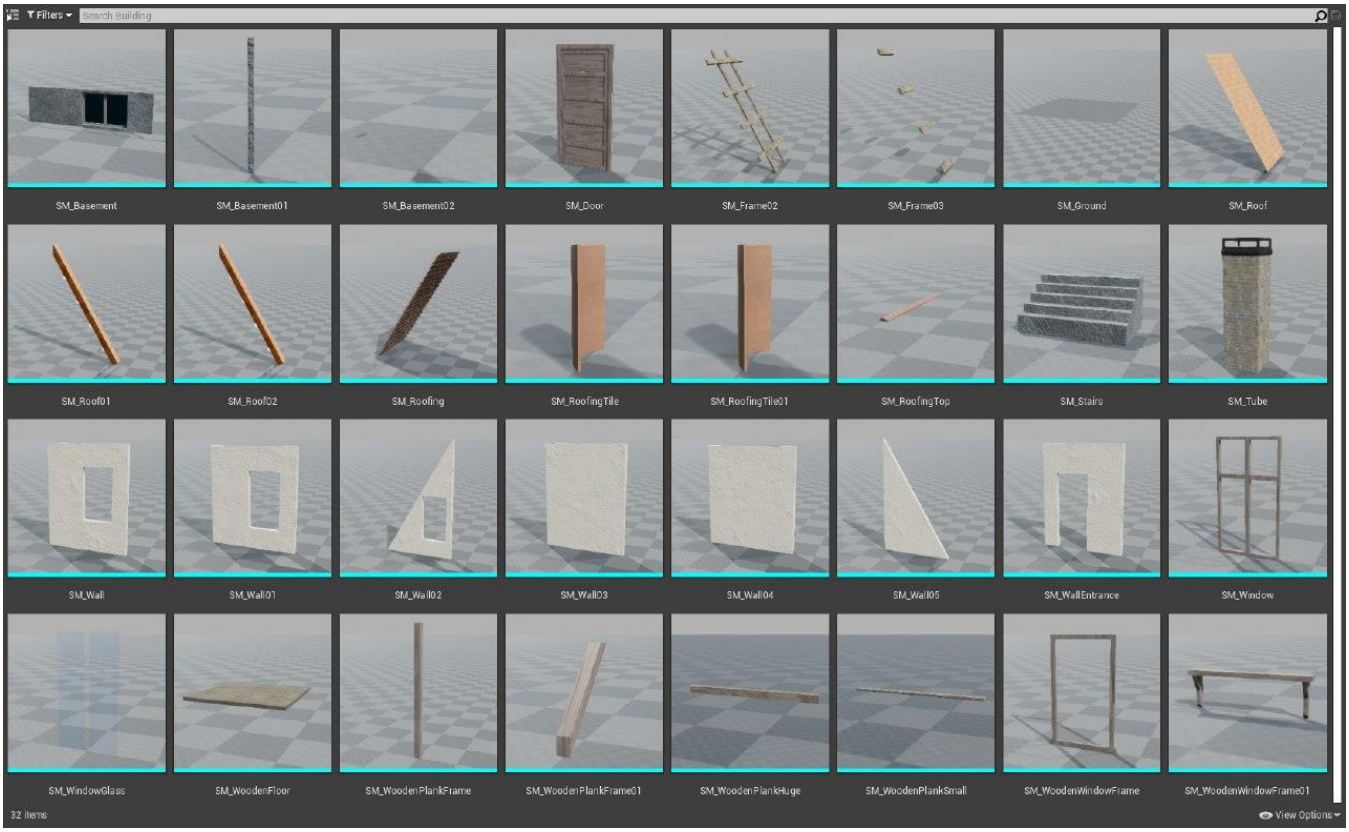


Рисунок 3.1 – Елементи конструкції одноповерхової будівлі в інтерфейсі Unreal Engine 4

Було сформовано повний список конструкційних частин моделі будівлі для моделювання.

3.3 Створення та нанесення текстур на моделі. Генерація масок.

Другим кроком стала побудова UV-розгортки. Ця розгортка поділена на секції квадратної форми (острівці), кожна з яких має свою назву щодо англійського алфавіту, наприклад острівець E6, рядок і стовпчик відповідно. Така розгортка має практичну значущість, коли існує багато частин однієї моделі і їх потрібно затекстурувати різними матеріалами. UV дозволяє:

- уникнути накладення текстур одна на одну;
- зберігати пропорційність розмірів текстур;

- коректно відображати тіні від попадання світла на об’єкт.

Третій крок - створення основної текстури та її масок, нанесення текстур на кожний елемент конструкції будівлі. Для вирішення цієї задачі було обрано дві програми - Substance Painter та Adobe Photoshop. Перш за все було взято стандартні текстури дерева, бетону та інших матеріалів із фотостоку з можливістю вільного використання (royalty free), важливо зауважити, що текстура мала бути безшовною для кращого нанесення на 3D-модель, без явно видимих артефактів. В текстуру були внесені правки в Photoshop та експортовано Substance Painter.

Substance Painter дозволяє із стандартної текстури згенерувати потрібні для рендерингу карти. Для цього проекту було згенеровано карту нормалей та ORM-карту.

Карта нормалей – карта, яка дозволяє візуально зробити імітацію неоднорідності поверхні стандартної текстури. Використовується для надання матеріалу об’єму.

3.4 Експорт моделі та всіх матеріалів в UE4

Після цього готові матеріали та модель будинку були експортовані у ігровий двигун Unreal Engine 4, де була згенерована карта світла (light map).

Карта свіла (англ. Light map) - карта, що дозволяє "запікти" (англ. bake) світло та тіні при обробці світла на сцені. Ця карта підходить для всіх статичних мешів на сцені.

Повертаючись до моделі, важливо сказати, що вона була перенесена в двигун в форматі .fbx (filmbox) та автоматично тріангульована, цю ж процедуру можна виконувати вручну в 3D-редакторі на етапі завершення моделювання та текстурування об’єкту.

Тріангуляція - розподіл багатогранних поверхонь на трикутники, при цьому одна і та сама поверхня може бути тріангульована різними способами, в залежності

від форми та кількості вершин площини. Вона допомагає уникнути виникненню артефактів та колізій, наприклад, при переміщенні об’єкта по тріангульованому ландшафту (англ. terrain) чи правильному падінню тіні від тріангульованого об’єкта.

3.5 Меші

Меші поділяють на статичні, скелетні та руйнуючі. В системі руйнування будинків використовують статичні та руйнуючі меші.

При експорті моделі, кожна з частин конструкції за замовчуванням є звичайним мешем, але при детонації усі модулі, окрім модуля "фундамент" переходять у стан "руйнуючий меш" (англ. destructible mesh)

3.5.1 Статичні меші

Instanced Static Mesh - це статичні меші, що входять в єдину групу, задля оптимізації пам'яті відеокарти, та кількості викликів низькорівневої функції DrawCall.

Кількість, тип мешу, а також відстань між цими мешами регулюється самим компонентом. При використанні цього компоненту, всі меші керуються одним. Тобто переміщуючи один меш, зміни застосовуються до всіх мешів в цій групі.

На рисунку 3.2 зображена група статичних мешів на прикладі дерев'яних ящиків.

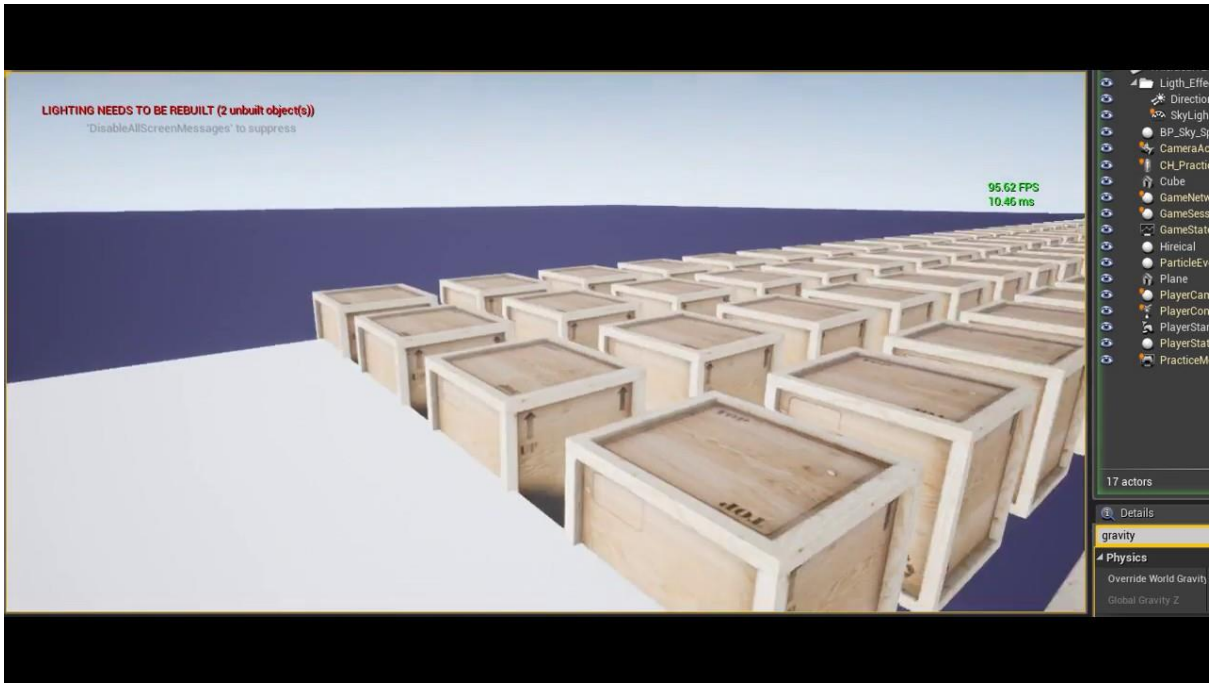


Рисунок 3.2 - Група статичних мешів

В системі руйнування будинків, цей компонент використовується для дублювання елементів черепиці. Це має велику значущість, коли потрібно використовувати дуже багато однакових елементів, в нашому випадку черепиці, задля спрощення та оптимізації руйнування. Тобто можна використати один такий компонент та встановити таку кількість мешів, яку потрібно для заповнення однієї сторони даху.

3.5.2 Динамічні меші

Destructible mesh - вбудована в двигун Unreal Engine 4 система руйнування об’єктів, яка дозволяє визначати та маніпулювати кількістю частинок уламків (фракцій) та формами уламків.

Руйнуючий меш може поділятися на задану кількість частин, яка генеруватися відповідно до користувацького ключа, та налаштовує кожну окрему частину руйнуючого мешу як невеликий, окремий, статичний меш. Може бути задіяний в системі деструкцій Unreal Engine 4.

Так як кожна частина руйнуючого мешу може виступати в ролі невеликого статичного мешу, то у кожної цієї частини є свій колайдер колізії. На рисунку 3.3 зображено колайдери колізій всіх частинок руйнуючого мешу двірного отвору.

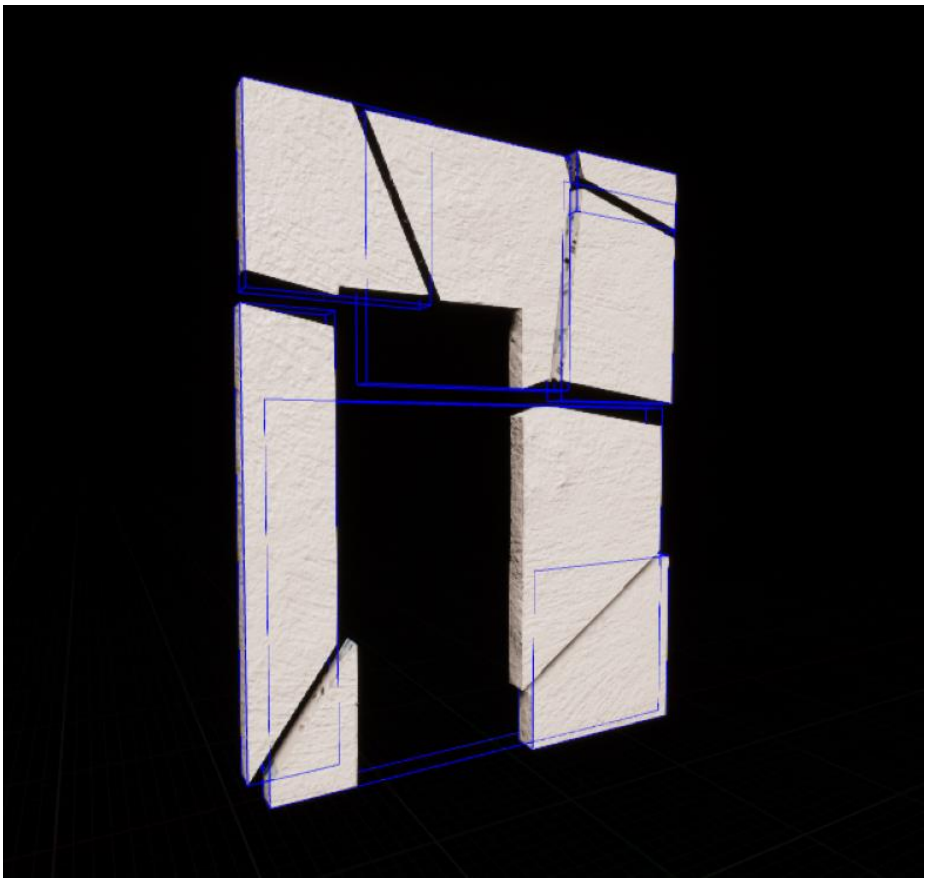


Рисунок 3.3 - Колайдери колізій руйнуючого мешу двірного отвору

За кількість фракцій відповідає параметр «fraction» в налаштуваннях мешу, а параметр «random seed» відповідає за генерування довільного числа в заданому розробником діапазоні чисел, задля того, щоб визначити положення розрізу об’єкту впливу.

Фракція – це частина чогось цілого. В даному випадку, фракція є однією частиною руйнуючого мешу.

Кожна конструкція будівлі поділена на N-ну кількість фракцій, в залежності від розміру об'єкта:

- велика модель, наприклад, стінка - поділена на 5 фракцій;

- маленька модель, наприклад, вікно - поділена на 3 фракції;
- виключенням є модель "даху" - вона поділена на 20 фракцій;
- та модуль "фундамент" який не ділиться на фракції, а є цільним і не відноситься до класу destructible mesh.

3.5.3 Налаштування колізій

Наступний крок - налаштування границь колізії. Так як всі складові будівлі мають чітку форму, майже завжди це форма паралелепіпеду, було вирішено використовувати автоматичне накладання колізії на кожен елемент, задля економії часу та ресурсів.

Колайдер колізії - це геометрична область, яка відображає границі фізичної взаємодії елемента з навколишнім середовищем під впливом фізичних законів зіткнення об’єктів. Колайдери використовуются для перевірок зіткнення одного об'єкту з іншим. В системі вибуху будинків, ця перевірка використовується для запобігання проникнення одного об’єкту в інший. А так як кількість частин будинку під час вибуху досягає більше сотні, то така перевірка є вкрай важливою.

На рисунку 3.4 продемонстровано примітивний колайдер колізії мешу, який знаходиться поверх двірного отвору.

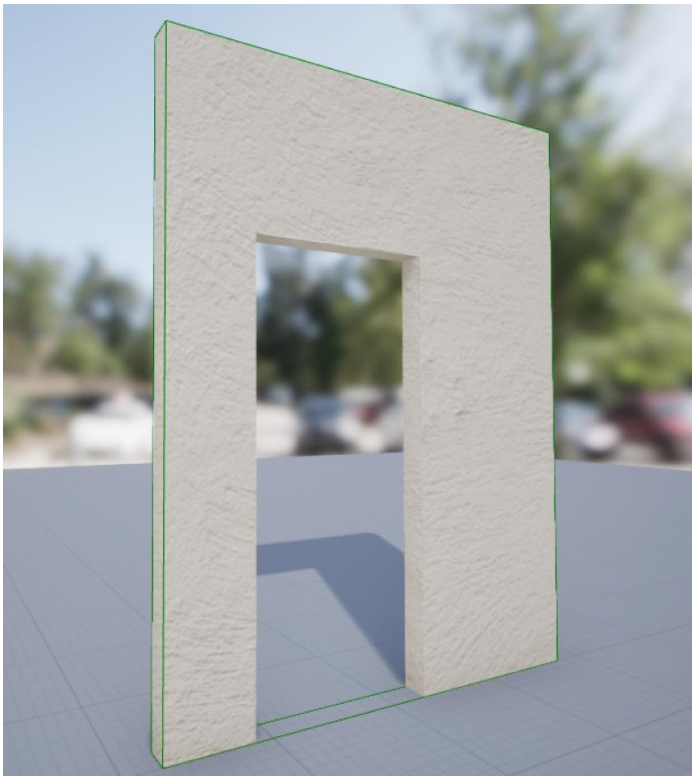


Рисунок 3.4 - Колайдер мешу двірного отвору

Кожна з частин конструкції будівлі має свій власний колайдер колізії, що запобігає їх проникненню одна в іншу та дозволяє фізично взаємодіяти з будь-якими іншими об’єктами сцени.

3.6 Додавання візуальних ефектів вибуху

Наступним етапом розробки стало, додавання візуальних ефектів детонації та ефектів при взаємодії колізії будинку з імпульсом радіальної сили точки вибуху. Потрібно було відтворити один основний візуальний ефект ("dust") з трьома компонентами:

- великі частини ґрунту;
- малі частини ґрунту;
- дим (імітація пилу).

Цього ефекту вдалось досягти за допомогою VFX. А саме за основу був взятий стандартний ефект вибуху, який знаходиться в початкових матеріалах двигуна UE4 та має ліцензію на вільне комерційне та некомерційне використання. Та він був модифікований за допомогою деяких налаштувань:

- було вилучено спалах вогню від вибуху;
- було збільшено радіус поширення диму; (розписати, де розташовані епіцентри поширення диму, та надати схему)
- було збільшено тривалість самого візуального ефекту;
- було додано дві перехресні текстури пилу та бруду, задля надання цьому спецефекту імітації об’ємності.

3.7 Система Blueprint

Головними блюпринтами системи є DestructibleBuilding_BP та DestructibleBuilding01_BP, об’єкти акторів, які містять готовий об’єкт будівлі з вбудованою логікою детонації та наступними параметрами контролю вибуху:

- радіус;
- сила імпульсу вибухової хвилі;
- сила удару вибухової хвилі.

Користувач системи має змогу встановити вибуховий пристрій в будівлю, або поблизу об’єкту впливу використовуючи блюпринти: DestructibleBuilding_BP \ DestructibleBuilding01_BP або Building_BP\Building01_BP, а також користувач системи має можливість змінювати та модифікувати існуючий Instancing_BP, який містить об’єкт даху.

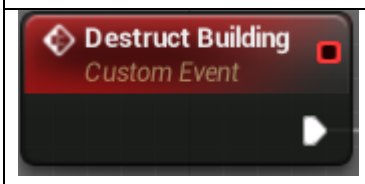
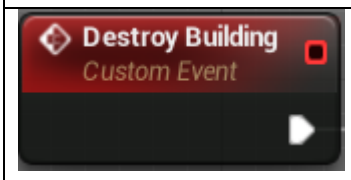
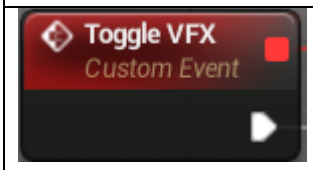
Якщо користувач заміняє початкову модель будівлі на свою та дах нової моделі має плоску форму, то немає потреби використовувати InstanceStaticMesh.

Система дозволяє легко змінювати наслідуванні компоненти даху (Roof’s Child Component’s) на будь-які статичні меші або інші компоненти.

3.7.1 Події системи

В таблиці 3.2 наведено список головних подій системи, які можна викликати, для керування системою.

Таблиця 3.2 – Список подій

Подія	Опис події
	**Destruct Building** - головна подія (event) в блюпринті **DestructibleBuilding_BP**, що відповідає за виклик детонації.
	**Destroy Building** - подія в **Building_BP** та **Building01_BP** викликають вибух базової частини будівлі в блюпринті **DestructibleBuilding_BP**.
	**Toggle VFX** - подія в **Building_BP**, що запускається з часом. Відповідає за активацію та дезактивацію відображення VFX-частинок (Particle System Component group).

Спеціальні події (custom event) – події, що створені користувачем та можуть бути викликані в як в одному блюпринті (локальне використання), так і в дочірніх блюпринтах (зовнішнє використання).

3.7.2 Властивості та параметри блюпринтів

Building_BP – блюпринт складається з повної конструкції будівлі, не враховуючи модель даху та VFX-часточки. Використовується як дочірній компонент (Child Actor Component) основного блюпринта **DestructibleBuilding_BP**. На рисунку 3.5 показано клас Building_BP з мініатюрою.

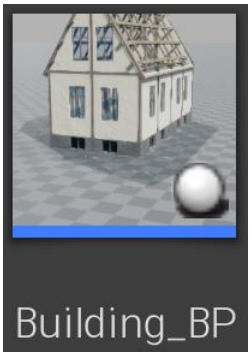


Рисунок 3.5 - Вигляд блюпринта **Building_BP** в інтерфейсі UE4

Компоненти групи:

- **Basement** – фундамент будівлі - статичний меш, який залишається на сцені після детонації;
- **Destructible** – групи руйнуючих мешів (destruction meshes) - динамічні меші, які знищуються в процесі детонації;
- **VFX** – спецефекти - система випромінювання частинок (particle system emitters) - спецефект, який активується в процесі детонації.

На рисунку 3.6 показано 3 групи компонентів блюпринта Building_BP, які до нього підключені.

Рисунок 3.6 - 3 групи компонентів блюпринта **Building_BP**

Building01_BP – блюпринт складається з повної конструкції будівлі, не враховуючи модель даху та VFX-часточки. Використовується як дочірній компонент (Child Actor Component) основного блюпринта **DestructibleBuilding_BP**. На рисунку 3.7 показано клас Building01_BP з мініатюрою.

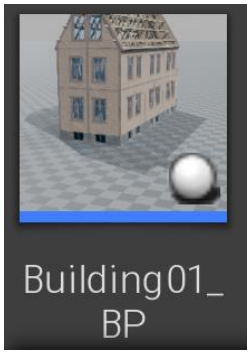


Рисунок 3.7 - Вигляд блюпринта **Building01_BP** в інтерфейсі UE4

Компоненти групи:

- **Basement** – фундамент будівлі - статичний меш, який залишається на сцені після детонації;
- **Destructible** – групи руйнуючих мешів (destruction meshes) - динамічні меші, які знищуються в процесі детонації;
- **VFX** – спецефекти - система випромінювання частинок (particle system emitters) - спецефект, який активується в процесі детонації.

На рисунку 3.8 показано 3 групи компонентів блюпринта Building01_BP, які до нього підключені.

Рисунок 3.8 - 3 групи компонентів блюпринта **Building01_BP**

DestructibleBuilding_BP – головний блюпринт, який містить одноповерховий будинок з фундаментом, дахом, VFX, вбудованою логікою вибуху, силою вибуху та параметрами контролю вибуху. На рисунку 3.9 показано клас DestructibleBuilding_BP, без мініатюри.

Рисунок 3.9 - Вигляд блюпринта **DestructibleBuilding_BP** в інтерфейсі UE4

Компоненти групи:

- **Building** – повна конструкція будівлі - дочірній компонент (the child actor component), який містить блюпринт Building_BP;
- **Roof1** – дах - дочірній компонент (child actor component), який містить блюпринт Instancing_BP з односторонньою черепицею. Може бути замінений будь-яким іншим наслідуваним об’єктом або мешем;
- **FloorFirst** – перший поверх - дочірній компонент (child actor component), який містить блюпринт Instancing_BP з об’єктом фундаменту/панеллю першого поверха. Може бути замінений будь-яким іншим дочірнім об’єктом або мешем;
- **RoofTop** – дочірній компонент (child actor component), який містить блюпринт Instancing_BP з верхівкою даху. Може бути замінений будь-яким іншим дочірнім об’єктом або мешем;
- **FloorLast** – панель останнього поверху - дочірній компонент (child actor component), який містить блюпринт Instancing_BP з панеллю останнього

поверху. Може бути замінений будь-яким іншим дочірнім об’єктом або мешем;

- **Roof2** – дах - дочірній компонент (child actor component), який містить блюпринт Instancing_BP з односторонньою черепицею. Може бути замінений будь-яким іншим дочірнім об’єктом або мешем;
- **Destructible** – групи руйнуючих мешів (destructible meshes) - динамічні меші, які знищуються в процесі детонації. Можуть бути замінені будь-якими іншими руйнівними мешами;
- **Stacks** – містить групу статичних мешів зі сміттям, який здіймається під час процесу детонації;
- **RadialForce** – радіальна сила - компонент сили поштовху руйнівних мешів відносно епіцентру вибуху.

На рисунку 3.10 показано групи компонентів блюпринта DestructibleBuilding_BP, які до нього підключені.

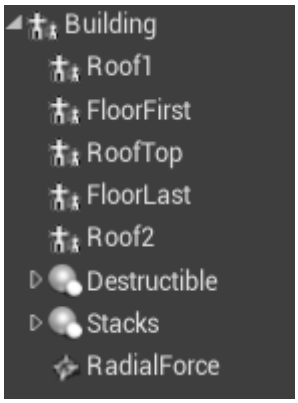


Рисунок 3.10 - Групи компонентів блюпринта **DestructibleBuilding_BP**

Змінні та їх значення:

- **Is Destructed** – булева змінна, яка вказує на то, чи будівля вже зруйнована. Значення true - зруйнована, значення false - ціла;
- **Radius** – змінна, яка приймає значення радіусу впливу радіальної сили (вибухова хвиля);

- **Impulse Strength** – змінна, яка приймає значення сили імпульсу радіальної сили;
- **Force Strength** – змінна, яка приймає значення сили імпульсу радіальної сили;

На рисунку 3.11 показано інтерфейс налаштувань параметрів вибуху блюпринта DestructibleBuilding_BP.

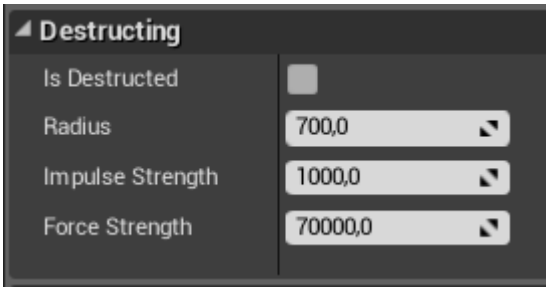


Рисунок 3.11 - Інтерфейс налаштувань параметрів вибуху блюпринта **DestructibleBuilding_BP**

DestructibleBuilding01_BP – головний блюпринт, який містить в собі одноповерховий будинок з фундаментом, дахом, VFX, вбудованою логікою вибуху, силою вибуху та параметрами контролю вибуху.

На рисунку 3.12 показано клас DestructibleBuilding01_BP, без мініатюри.

Рисунок 3.12 - Вигляд блюпринта **DestructibleBuilding01_BP** в інтерфейсі UE4

Компоненти групи:

- **Building** – повна конструкція будівлі - дочірній компонент (the child actor component), який містить блюпринт Building_BP;
- **Roof1** – дах - дочірній компонент (child actor component), який містить блюпринт Instancing_BP з односторонньою черепицею. Може бути замінений будь-яким іншим наслідуванним об’єктом або мешем;
- **FloorFirst** – перший поверх - дочірній компонент (child actor component), який містить блюпринт Instancing_BP з об’єктом фундаменту/панелью першого поверха. Може бути замінений будь-яким іншим дочірнім об’єктом або мешем;
- **RoofTop** – дочірній компонент (child actor component), який містить блюпринт Instancing_BP з верхівкою даху. Може бути замінений будь-яким іншим дочірнім об’єктом або мешем;
- **FloorLast** – панель останнього поверху - дочірній компонент (child actor component), який містить блюпринт Instancing_BP з панелью останнього поверху. Може бути замінений будь-яким іншим дочірнім об’єктом або мешем;
- **Roof2** – дах - дочірній компонент (child actor component), який містить блюпринт Instancing_BP з односторонньою черепицею. Може бути замінений будь-яким іншим дочірнім об’єктом або мешем;
- **Destructible** – групи руйнуючих мешів (destructible meshes) - динамічні меші, які знищуються в процесі детонації. Можуть бути замінені будь-якими іншими руйнівними мешами;
- **Stacks** – містить групу статичних мешів зі сміттям, який здіймається під час процесу детонації;
- **RadialForce** – радіальна сила - компонент сили поштовху руйнівних мешів відносно епіцентру вибуху;
- **FloorMiddle** – перший поверх - наслідуваний компонент (the child actor component), який містить блюпринт Instancing_BP з об’єктом панеллю

другого, третього, четвертого та інших поверхів. Може бути замінений будь-яким іншим наслідуваним об’єктом або мешем.

Instancing_BP – блюпринт-система, яка містить InstancedStaticMesh з логікою спорудження черепиці даху. На рисунку 3.13 показано клас Instancing_BP, без мініатюри, так як цей компонент є службовим, і не має мешу всередині. Меш для його роботи розміщується уже в створений ним клас об’єкту.

Рисунок 3.13 - Вигляд блюпринта **Instancing_BP** в інтерфейсі UE4

3.7.3 Компоненти мешів в блюпринтах. Їх налаштування.

В блюпринті **Building_BP\Building01_BP,** вкладка "Basement", містить статичний меш, який залишається статичним і під час детонації.

На рисунку 3.14 показано вкладку "Basement", який містить всі меші фундаменту, що залишаються після вибуху.

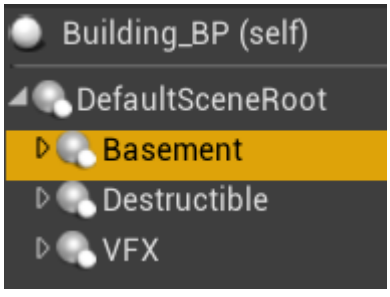


Рисунок 3.14 - Вигляд та положення вкладки "Basement" в інтерфейсі UE4

В блюпринті **Building_BP\Building01_BP,** вкладка "Destructible", містить руйнівний меш, який знищується під час вибуху.

- Розміщено усі вікна (меші зі склом) у підкатегорії "Windows";
- Розміщено усі опорні балки у підкатегорії "Frame";
- Будь-які інші руйнівні меші, крім підлоги та даху, розміщені безпосередньо в групі "Destructible".

На рисунку 3.15 показано вкладку "Destructible", який містить всі меші руйнівних мешів, що будуть задіяні у вибусі.

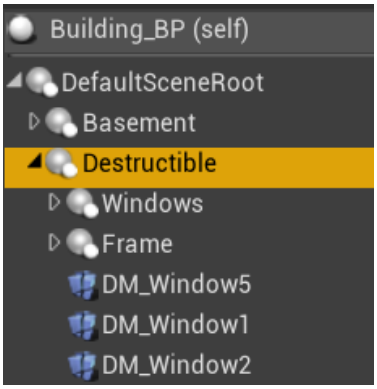


Рисунок 3.15 - Вигляд та положення вкладки "Destructible" в інтерфейсі UE4

В блюпринті **DestructibleBuilding_BP\DestructibleBuilding01_BP** поділено статичний меш "дах" на компоненти: Roof1, RoofTop, Roof2, дочірнім актором (child actor). Змінити меш даху можна, змінивши шаблон Instancing_BP або (якщо користувацький меш даху є цільним), замінивши компонент дочірнього актора будь-якими іншими статичними мешами з оновленими посиланнями на них в логіці блюпринта, якщо це необхідно.

На рисунку 3.16 показано компоненти Roof1, RoofTop, Roof2, які містять меші даху та налаштовані як InstancedStaticMesh

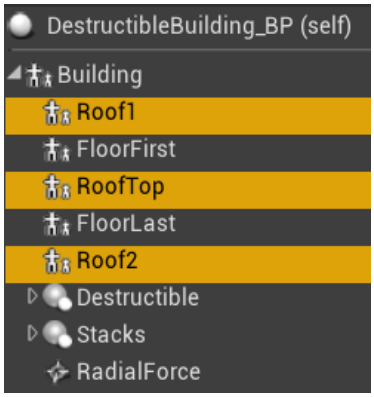


Рисунок 3.16 - Вигляд компонентів Roof1, RoofTop, Roof2 в інтерфейсі UE4

В блюпринті **DestructibleBuilding_BP \ DestructibleBuilding01_BP** міститься статичний меш підлоги в компонентах для дочірних елементів: FloorFirst, FloorLast (та FloorMiddle в DestructibleBuilding01_BP). Замінити меш підлоги можливо, змінивши шаблон Instancing_BP або (якщо користувацький меш підлоги є цільним), замінивши компонент дочірнього актора будь-якими іншими статичними мешами з оновленими посиланнями на них в логіці блюпринта, якщо це необхідно.

На рисунку 3.17 показано компоненти FloorFirst та FloorLast які містять всі меші підлоги включно с деструктивними мешами, що будуть задіяні у вибусі.

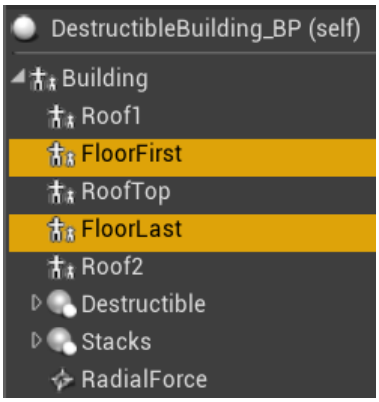


Рисунок 3.17 - Вигляд компонентів FloorFirst, FloorLast в інтерфейсі UE4

В блюпринті **DestructibleBuilding_BP \ DestructibleBuilding01_BP**, у вкладці "Destructible", відсортовані руйнуючі меші від статичних (що містять підлогу та дах) за категоріями:

- усі поверхи розміщені в підкатегорії "Floor";
- весь дах розміщенний у підкатегоріях "Roof_D" та "RoofTop_D".

На рисунку 3.18 показано вкладку "Destructible", який містить всі меші руйнівних мешів, що будуть задіяні у вибусі.

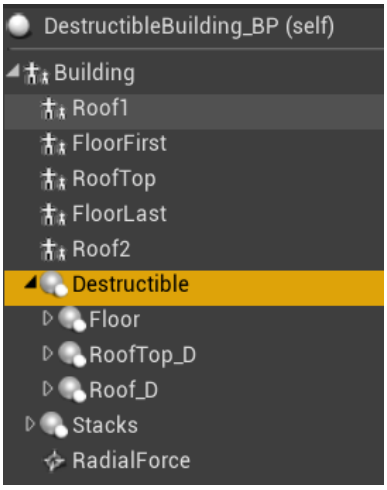


Рисунок 3.18 - Вигляд та положення вкладки "Destructible" в інтерфейсі UE4

В блюпринті **DestructibleBuilding_BP \ DestructibleBuilding01_BP** модна переміщати джерела VFX-частинок, щоб відрегулювати їх розташування, якщо користувацька будівля відрізняється за розміром від стандартної, представленої в системі. Користувач має можливість додавати, редагувати та видаляти деякі стеки.

Ці меші за замовчуванням є невидимими. Для зміни їх місцеположення потрібно зробити усі меші видимими, відрегулювати координати та сховати їх знову.

На рисунку 3.19 показано вкладку "Stacks", які містять усі статичні меші бруду, що будуть підійматися під час вибуху.

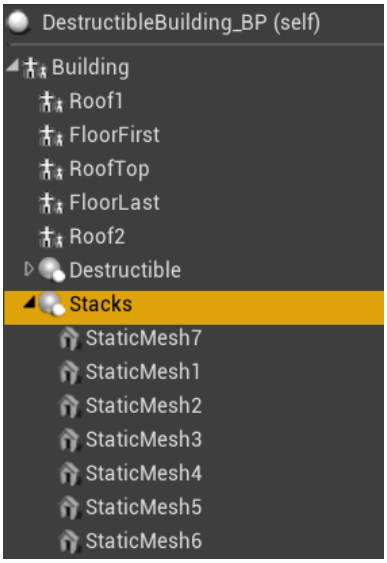


Рисунок 3.19 - Вигляд та положення вкладки "Stacks" в інтерфейсі UE4

В блюпринті **Building_BP \ Building01_BP** можна регулювати розташування VFX-частинок вибуху, щоб зробити візуальний ефект більш реалістичним. Якщо користувацька будівля має інший розмір, порівняно з представленою у системі, можна видалити деякі випромінювачі частинок або додати нові. У частинках включена автоматична активація за замовчуванням. Для зміни місцезнаходження джерела потрібно увімкнути автоматичну активацію для всіх джерел, відрегулювати його місцеположення та знову вимкнути автоматичну активацію.

На рисунку 3.20 показано вкладку VFX, які містять усі візуальні частини, що активуються під час вибуху.

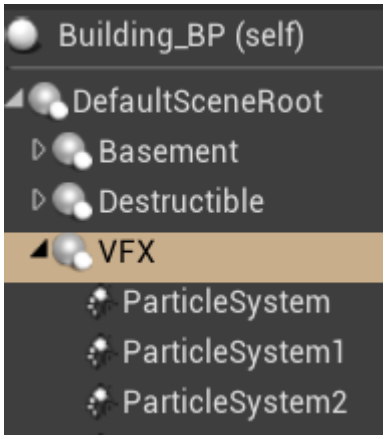


Рисунок 3.20 - Вигляд та положення вкладки VFX в інтерфейсі UE4

Наостанок, потрібно відрегулювати значення радіальної сили в категорії "Destruction" в деталях блюпринта **DestructibleBuilding_BP** та/або **DestructibleBuilding01_BP**.

На рисунку 3.21 показано вкладку властивостей об'єкту DestructibleBuilding, які містять усі налаштування цього об’єкту.

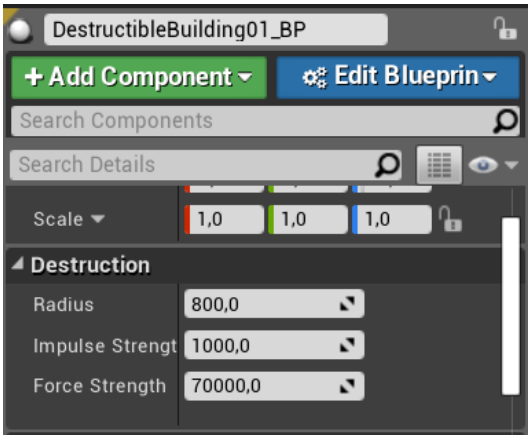


Рисунок 3.21 - Вигляд вкладки властивостей об'єкту DestructibleBuilding в інтерфейсі UE4

В розділі було наведено всі компоненти, що були використані у програмному плагіні: модель, типи мешів, колайдер колізії, логіка блюпринтів.

3.8 Діаграма класів

Програмний застосунок містить в собі п’ять класів: Building_BP, Building01_BP, DestructibleBuilding_BP, DestructibleBuilding01_BP та Instansing_BP.

Класи Building_BP та Building01_BP містять в собі всі основні елементи конструкції будівлі: фундамент, всі руйнівні об’єкти, випромінювачі VFX-часточок. Також в клас закладено логіку, що виводить зі сплячого (неактивного) режиму всі руйнівні меші та активує спецефекти. Клас Building_BP – одноповерховий будинок, клас Building01_BP – двоповерховий будинок.

Класи DestructibleBuilding_BP та DestructibleBuilding01_BP містять в собі об’єкти класів Building_BP та Building01_BP, відповідно, та Instansing_BP, як компоненти цього класу. В два цих класи закладено логіку запуску процесу руйнування будівель, саме: запуск детонації основи будівлі компоненту Building_BP/Building01_BP, видалення статичних мешів, активація видимості всіх

руйнівних мешів, включення фізичної симуляції, активація радіальної сили, що починає процес вибуху, поява мешів з уламками по периметру фундаменту.

Клас Instancing_BP містить логіку задання кількості об’єктів для групи статичних мешів (Instanced Static Mesh), завдяки якій задається один об’єкт черепиці та вказується розмір поля, на якому має бути вистелена черепиця, в умовних одиницях. Логіка класу розраховує потрібну кількість черепиць за для заповнення заданого поля.

На рисунку 3.22 відображені всі класи плагіну та їх взаємозв’язок між собою.

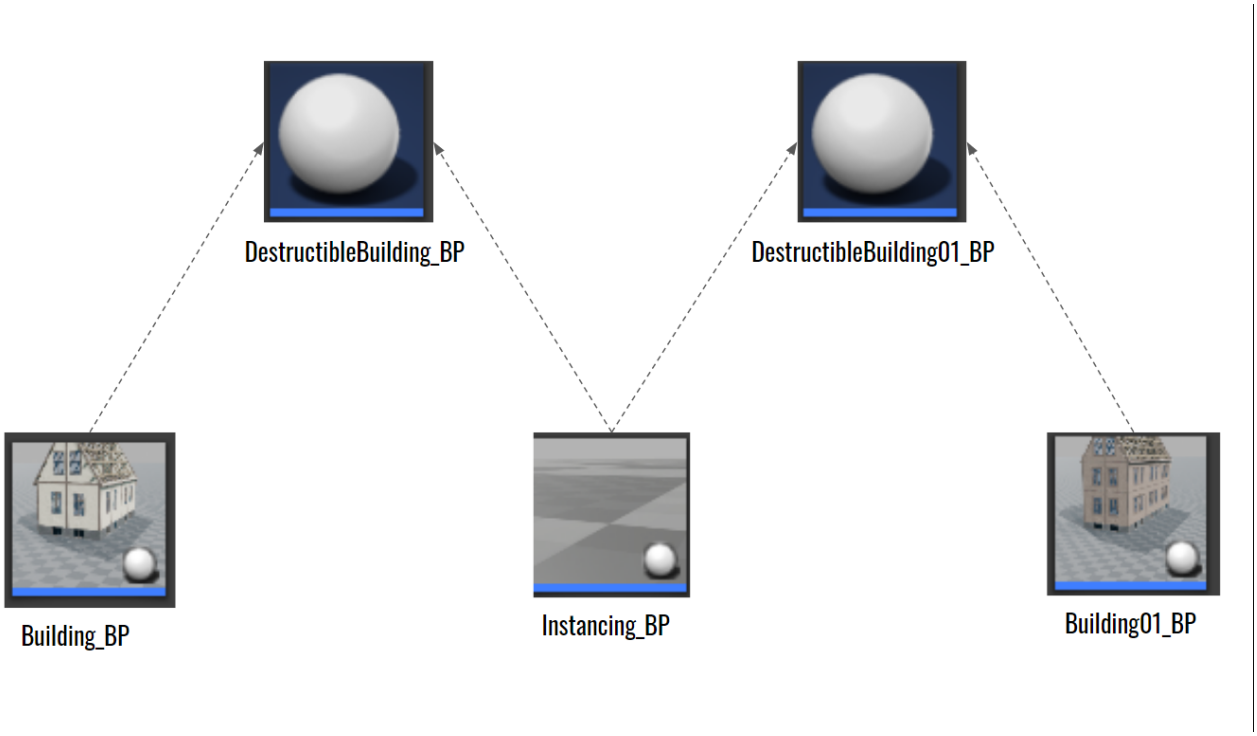


Рисунок 3.22 - діаграма класів системи

Кожний клас даного програмного забезпечення відстроєний за допомогою логіки мови Blueprint.

4 Обчислювальні експерименти

В цьому розділі приведені дані щодо проведених обчислювальних експериментів системи з метою оптимізації навантаження на відеокарту під час процесу детонації.

Головним завданням експериментів було підвищити кількість кадрів за секунду (Frames per second) після запуску функції детонації, тим самим зменшивши різницю відображення кадрів до початку вибуху і під час цього процесу.

В першому експерименті початковий FPS до запуску функції детонації становив 120fps. Модель черепиці була максимально деталізована - була використана вискокоплігональна модель кожної черепиці (high polygonal).

Результатами першого експерименту стало зниження частоти кадрів зі 120 до 30fps. Що стало значним зниженням та призвело до гальмування системи. Основною проблемою став факт переходу мешу кожної черепиці зі статичного стані в динамічний. А це більш ніж 100 елементів черепиці поділених на 4 фракції, що в сумі викликає навантаження на графічну карту в 3-4 рази.

В другому експерименті, взявши до уваги особливості руйнування даху, було вирішено використати замість високополігональних моделей їх більш примітивні - низькополігональні (low polygonal) аналоги. Весь дах був замінений новими моделями черепиці.

Підсумок другого експерименту дали очікувані результати: частота кадрів збільшилась, але збільшилась незначно - з 30fps в першому експерименті до 50fps в другому.

В третьому експерименті було вирішено протестувати модель даху, яка замість форми черепиць мала форму простої фігури - паралелепіпеду, задля досягнення максимального спрощення.

Результати третього випробування були наступними: частота кадрів збільшилась на 10fps відносно попереднього випробування, досягнувши позначки в 60fps.

В четвертому експерименті було поставлено питання підвищення кадрів ще на 20-40 одиниць, так як плагін надалі має потенціал бути пристосованим до мобільних ігор. З останнього експерименту можна зробити висновок, що проблема зниження частоти кадрів під час детонації знаходилась в більшій мірі не в моделі даху. Проблема виникала в самому блюпринті при виклиці функції детонації. Цей блюпринт містив багато компонентів конструкції будівлі - внутрішніх опорних балок. Ці елементи значно знижували fps, проте не мали високої значущості для симуляції. Ці компоненти були видалені, що призвело до підвищення частоти кадрів за секунду до 105fps. Важливо зазначити, що в цьому експерименті також були використані матеріали з різним рівнем деталізації (LOD - level of design) та встановленим значенням автоматично, що дозволяє двигуну самостійно оптимізувати якість текстур та кількість полігонів конкретних моделей в залежності від відстані від камери до об’єкту.

На рисунку 4.1 відображені показники відеокарти при початковому стані об’єктів впливу. На сцені встановлено 2 будівлі та 2 точки детонації в кожній з будівель.

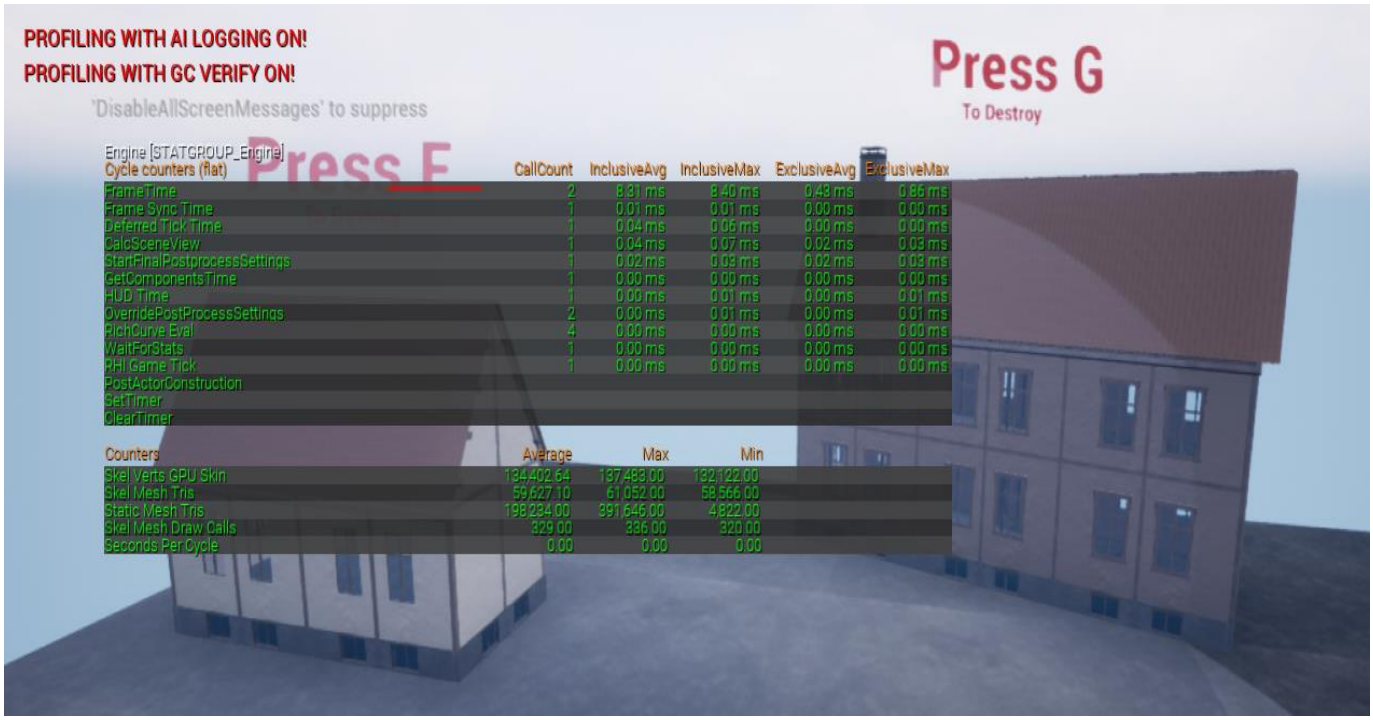


Рисунок 4.1 – Показники відеокарти на 8.31 мілісекунді в стані спокою

На рисунку 4.2 відображені показники відеокарти при детонації одноповерхового будинку.

Рисунок 4.2 - Показники відеокарти на 12.12 мілісекунді з максимальним значенням та з середнім значенням на 8.56 мілісекунді

На рисунку 4.3 відображені показники відеокарти при детонації двоповерхового будинку. Важливо зауважити, що даний експеримент був поставлений відносно початкового стану сцени, коли на сцені були присутні обидві будівлі.

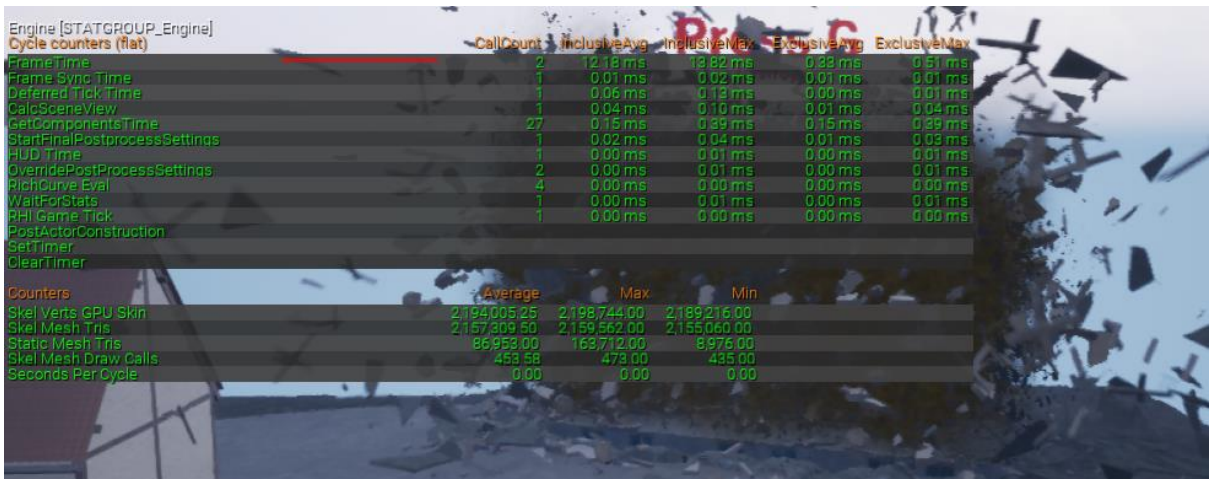


Рисунок 4.3 - Показники відеокарти на 13.82 мілісекунді з максимальним значенням та з середнім значенням на 12.18 мілісекунді

Ряд проведених експериментів з оптимізації графіки та відповідних блюпринтів дали значний приріст в кількості кадрів за секунду, що підвищилась на 75 одиниць відносно першого випробування.

5 МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

В цьому розділі приведені системні вимоги для роботи з додатком та сценарії роботи користувача з ним.

5.1 Інсталяція та системні вимоги

Для забезпечення коректної та безвідмовної роботи програмної системи детонації будівлі потрібно дотримуватись наступних вимог.

Мінімальні вимоги до пристрою:

- станом на версію 4.24, пристрої Android, які використовують OpenGL ES 3.1, використовує стандартну специфікацією для Unreal Engine;
- усі пристрої, що відповідають цим вимогам, сумісні з проектами UE4;
- крім того, ці функції доступні на пристроях ES 3.2 за замовчуванням, і всі пристрої ES 3.2 повинні бути сумісні з проектами UE4.

Мінімально підтримувані GPU:

Наступні графічні процесори відповідають вищезазначеним вимогам щодо ES3.1 версії 4.25:

- Adreno 4xx;
- Mali T6xx та новіші версії;
- Mali G71 та новіші версії;
- PowerVR Rogue G6100.

5.2 Інструкція використання системи з користувацькою 3D-моделью

Алгоритм імплементації користувацької будівлі:

1) Розробити модульний 3D-об`єкт в редакторі;
2) Згрупувати меші за категоріями компонентів системи наступним чином:
    - фундамент;
    - руйнівні елементи конструкції;
    - вікна;
    - стіни;
    - тощо.
3) Оновити посилання на нові меші в логіці блюпринт-системи;
4) Якщо розмір нової моделі не збігається з розміром мешів за замовчуванням, потрібно перемістити джерела VFX, щоб виправити їх позиції;
5) Встановити значення радіалої сили та інші параметрів в компонентах блюпринт-системи , щоб зробити руйнування більш реалістичним.

5.3 Демонстрація роботи

На рисунках 5.1-5.10 представлені стани кожної з будівель в конкретний проміжок часу до їх повного руйнування.

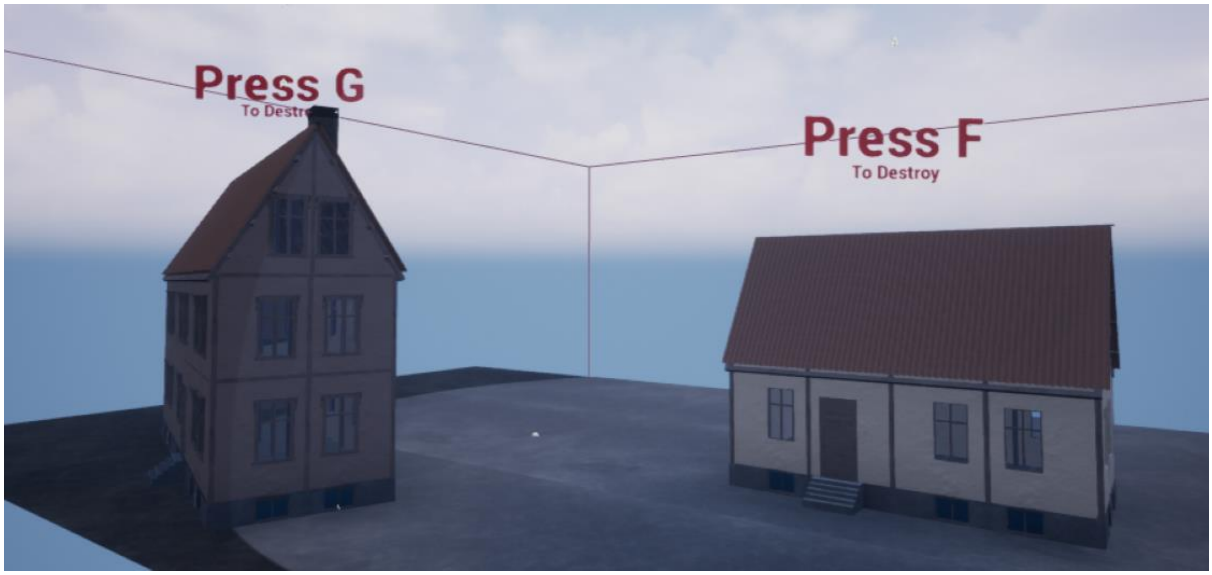


Рисунок 5.1 - Будівлі в початковому стані

Рисунок 5.2 - Одноповерховий будинок на другій секунді після початку вибуху

Рисунок 5.3 - Одноповерховий будинок на четвертій секунді після вибуху

Рисунок 5.4 - Одноповерховий будинок на п’ятій секунді після початку вибуху

Рисунок 5.5 - Повністю зруйнований одноповерховий будинок

Рисунок 5.6 - Двоповерховий будинок на початковому етапі

Рисунок 5.7 - Двоповерховий будинок на другій секунді після початку детонації

Рисунок 5.8 - Двоповерховий будинок на четвертій секунді після детонації

Рисунок 5.9 - Двоповерховий будинок на п’ятій секунді після початку вибуху

Рисунок 5.10 - Повністю зруйнований двоповерховий будинок

Продемонстровано результати візуалізації процесу детонації будівель з використанням спеціальних ефектів та однією точкою вибуху.

ВИСНОВКИ

В ході виконання даної роботи було розроблено плагін симуляції детонації будівлі, яка може використовуватись як один із модулів програмного забезпечення користувача за рахунок простої інтеграції плагіну в користувацьку систему, а також можливості заміни моделей та налаштувань вибуху.

Програму можна використовувати для імітації спецефектів в комп’ютерних та мобільних іграх.

1) Обрано такі програмні засоби створюваного плагіну: редактор Blender 3D, графічні редактори Substance Painter та Adobe Photohop та двигун Unreal Engine 4;
2) Розроблено 3D-моделі елементів конструкції будівлі;
3) Експортовано моделі та матеріали у двигун. Встановлені та налаштовані на сцені;
4) Статичні меші переведено в стан динамічних для реалізації вибуху.
5) Встановлено колайдери колізії для кожної окремої фракції елемента конструкції будівлі;
6) Створено спецефекти вибуху;
7) Розроблено логіка реалізації процесу детонації
8) З’єднано всі компоненти в єдиний плагін Destructible Buildings Systems
9) Проведено обчислювальні експерименти з оптимізації швидкості відображення, за результатами яких кількість кадрів за секунду підвищилась з 30 до 105fps.

Для роботи з даним програмним забезпеченням необхідний лише комп’ютер середньої потужності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tom Shannon, Unreal Engine 4 for Design Visualization Developing Stunning Interactive Visualizations, Animations, and Renderings. 2018. № 1 — 362с.
2. Unreal Engine 4 Documentation [Електронний ресурс] — Режим доступу: https://docs.unrealengine.com/
3. Сайт Udemy // Unreal Engine 4 - VFX for Games - Beginner to Intermediate [Електронний ресурс] — Режим доступу: https://www.udemy.com/course/ue4-vfx-for-games-beginner-to-intermediate/

ДОДАТОК А

3D-моделювання будівель

Специфікація

УКР.КПІ ім. Ігоря СІКОРСЬКОГО_ТР-62150_20Б

Аркушів 1

Київ 2020

Позначення	Найменування	Примітки
Документація		
УКР.КПІ ім. Ігоря СІКОРСЬКОГО_ТР-62150_20Б	ТР-62 Фахріян Денис Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.КПІ ім. Ігоря СІКОРСЬКОГО_ТР-62150_20Б-1	Building_BP.uasset	Модель одноповерхової будівлі (всіх її статичних компонентів)
УКР.КПІ ім. Ігоря СІКОРСЬКОГО_ТР-62150_20Б-2	Building01_BP.uasset	Модель двоповерхової будівлі (всіх її статичних компонентів)
УКР.КПІ ім. Ігоря СІКОРСЬКОГО_ТР-62150_20Б-3	DestructibleBuilding_BP.uasset	Модель одноповерхової будівлі (всіх її динамічних компонентів)
УКР.КПІ ім. Ігоря СІКОРСЬКОГО_ТР-62150_20Б-4	DestructibleBuilding01_BP.uasset	Модель двоповерхової будівлі (всіх її динамічних компонентів)
УКР.КПІ ім. Ігоря СІКОРСЬКОГО_ТР-62150_20Б-5	Instancing_BP.uasset	Модель симуляції вибуху зі спецефектами

ДОДАТОК Б

3D-моделювання будівель

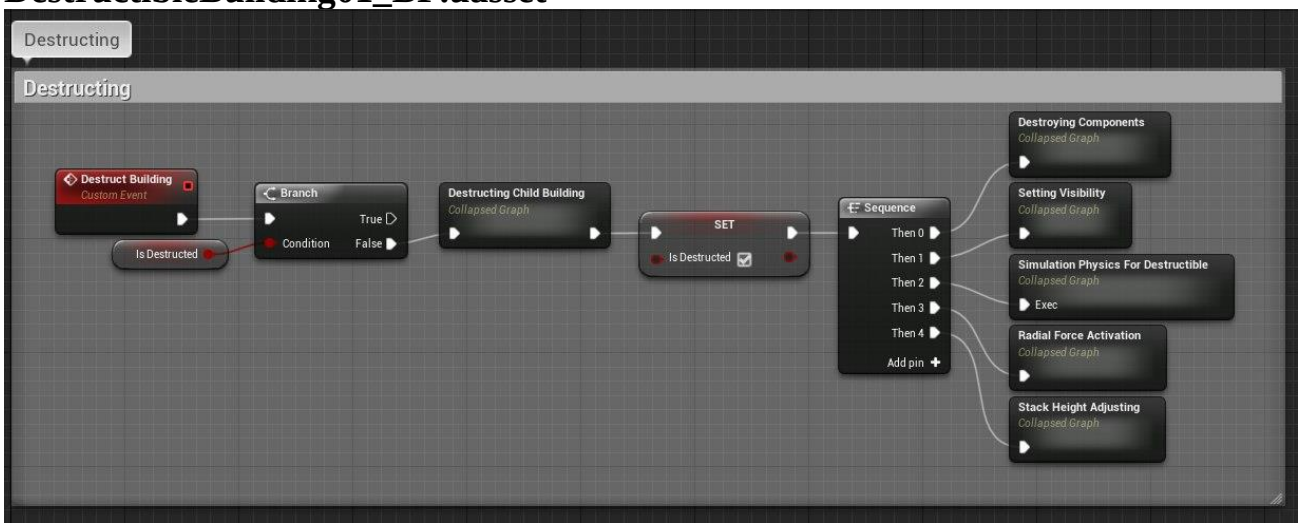
Текст програми

УКР.КПІ ім. Ігоря СІКОРСЬКОГО_ТР-62150_20Б

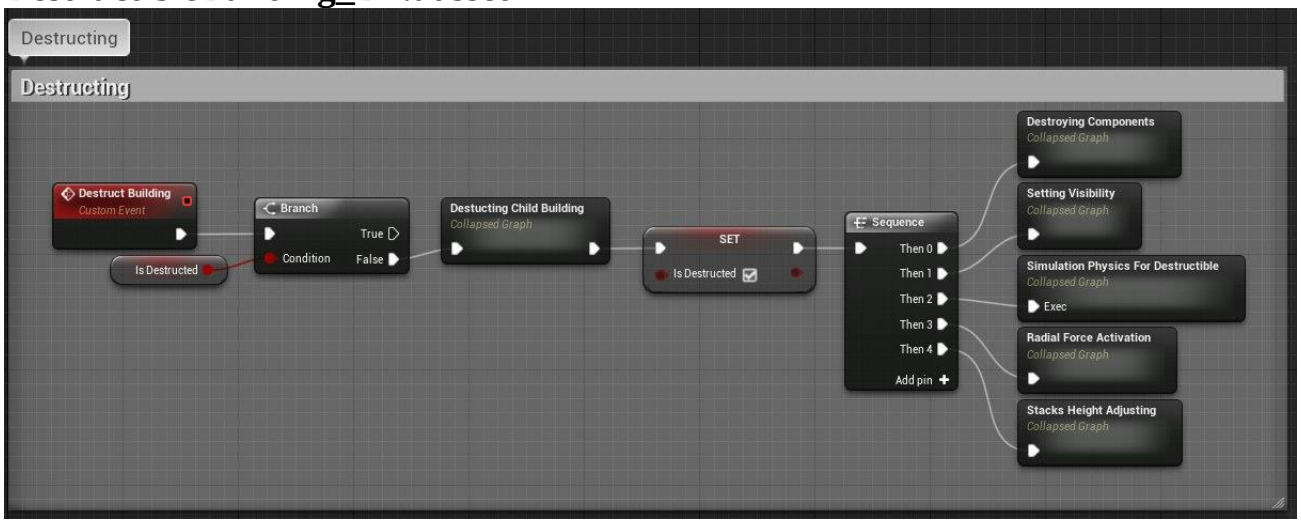
Аркушів 2

Київ 2020

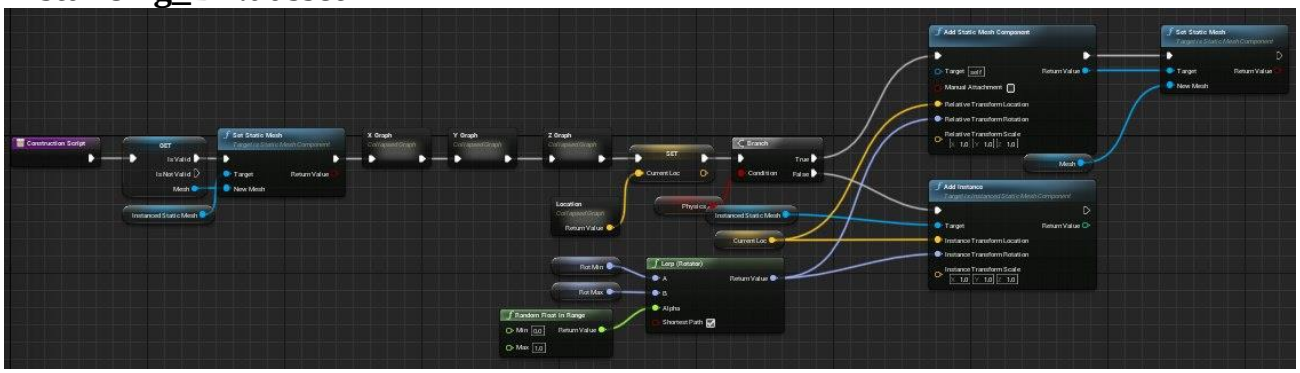
DestructibleBuilding01_BP.uasset



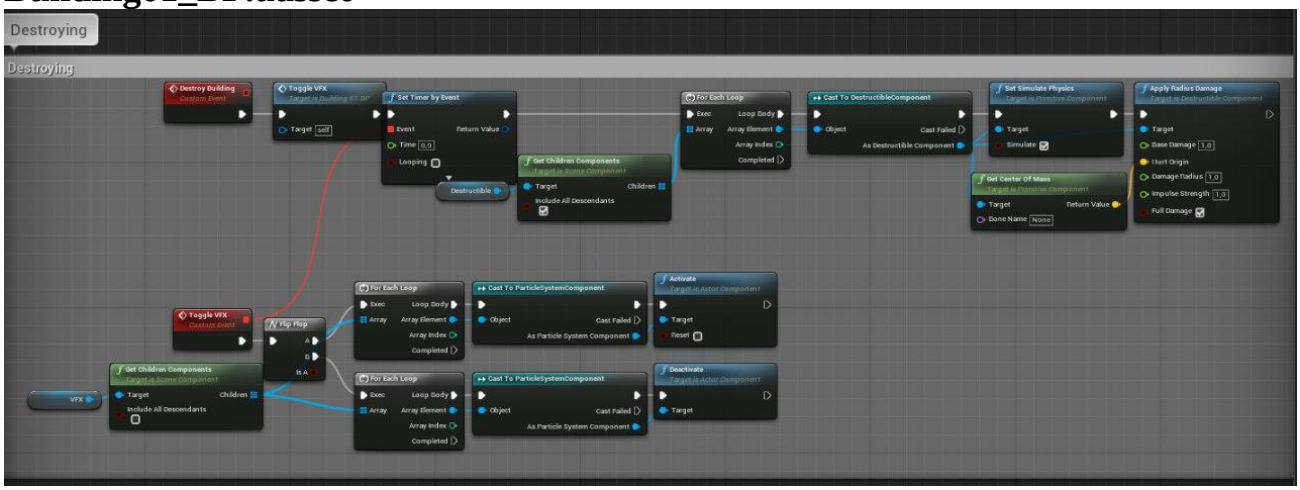
DestructibleBuilding_BP.uasset



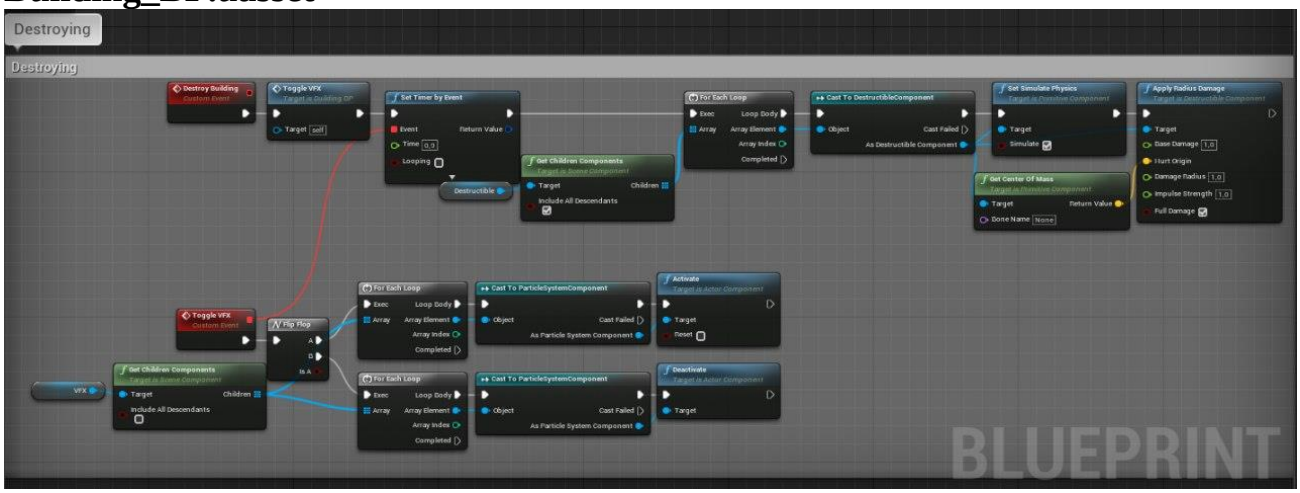
Instancing_BP.uasset



Building01_BP.uasset



Building_BP.uasset



ДОДАТОК В

3D-моделювання будівель

Опис програмного коду

УКР.КПІ ім. Ігоря СІКОРСЬКОГО_ТР-62150_20Б

Аркушів 2

Київ 2020

В.1 Файл моделі Building_BP.uasset

Файл містить модель одноповерхової будівлі без даху та підлоги та містить логіку вибуху базових частин об’єкту, яка відтворює VFX-ефекти під час вибуху і перетворює статичні меші на руйнівні. Саму візуальну модель можна змінити, змінюючи всі статичні та руйнівні в компонентах цього блюпринта і налаштовуючи їх за наведеним шаблоном.

В.2 Файл моделі Building01_BP.uasset

Файл містить модель двоповерхової будівлі без даху та підлоги нижнього та верхнього поверхів, містить логіку вибуху базових частин дому, яка відтворює VFX-ефекти під час вибуху і перетворює статичні меші на руйнівні. Саму візуальну модель можна змінити, змінюючи всі статичні та руйнівні в компонентах цього блюпринта і налаштовуючи їх за наведеним шаблону.

В.3 Файл моделі Instancing_BP.uasset

Файл містить логіку налаштування Instanced Static Mesh, що налаштовує кількість мешів в залежності від вказаної довжини потрібної групи мешів. Саму візуальну модель статичного мешу можна змінити в налаштуваннях цього файлу. Також в цій моделі передбачені швидкі налаштування для мешів з не стандартними розмірами Scale, задля коректного групування мешів.

В.4 Файл моделі DestructibleBuilding_BP.uasset

Файл містить модель одноповерхової будівлі, даху, підлоги, невеликих скупчень уламків, які з’являються після вибуху, і головну логіку запуску вибуху. Саму візуальну модель підлоги та даху можна змінити, змінюючи всі відповідні

статичні та руйнівні меші в компонентах цього блюпринта і налаштовуючи їх за наданим шаблоном.

В.5 Файл інтерфейсу DestructibleBuilding01_BP.uasset

Файл містить модель двоповерхового будинку, даху, підлоги, невеликих скупчень уламків, які з’являються після вибуху, і головну логіку запуску вибуху. Саму візуальну модель підлоги та даху можна змінити, змінюючи всі відповідні статичні та руйнівні меші в компонентах цього блюпринта і налаштовуючи їх за наданим шаблоном.